UNITED STATES PATENT APPLICATION

of

Andrej Kocev

and

Samuel H. Duncan

and

Stephen Ho

for a

PROGRAMMABLE TUNING FOR FLOW CONTROL AND SUPPORT FOR CPU HOT PLUG

PROGRAMMABLE TUNING FOR FLOW CONTROL AND SUPPORT FOR CPU HOT PLUG

CROSS-REFERENCE TO RELATED APPLICATIONS

The present application claims priority from U.S. Provisional Patent Application Serial No. 60/229,830, which was filed on 08/31/00, by the present inventors among others for a Symmetrical Multiprocessor Computer System and which is hereby incorporated herein by reference.

BACKGROUND OF THE INVENTION

Field of the Invention

The present invention relates to symmetrical distributed multiprocessor computer system architectures, and more particularly to adapting the system and its resources to improve system performance.

Background Information

Distributed shared memory computer systems, such as symmetric multiprocessor (SMP) systems, support high performance application processing. Conventional SMP systems include a plurality of processors coupled together by a bus. One characteristic of SMP systems is that memory space is typically shared among all of the processors. That is, each processor accesses programs and data in the shared memory, and processors communicate with each other via that memory (e.g., through messages and status information left in shared address spaces). In some SMP systems, the processors may also be able to exchange signals directly. One or more operating systems are typically stored in the shared memory. These operating systems control the distribution of processes or threads among the various processors. The operating system kernels may execute on any processor, and may even execute in parallel. By allowing many different processors to execute different processes or threads simultaneously, the execution speed of a given application may be greatly increased.

Fig. 1 is a block diagram of a conventional SMP system 100. System 100 includes a plurality of processors 102a-e, each connected to a system bus 104. A memory 106 and an input/output (I/O) bridge 108 are also connected to the system bus 104. The I/O bridge 108 is also coupled to one or more I/O busses 110a-c. The I/O bridge 108 basically provides a "bridging" function between the system bus 104 and the I/O busses 110a-c. Various I/O devices 112, such as disk drives, data collection devices, keyboards, CD-ROM drives, etc., may be attached to the I/O busses 110a-c. Each processor 102a-e can access memory 106 and/or various input/output devices 112 via the system bus 104. Each processor 102a-e has at least one level of cache memory 114a-e that is private to the respective processor 102a-e.

In large multiprocessor computer systems, the manner in which system resources are allocated can significantly affect performance. High flexibility is also desired so as to increase the number of uses to which the system may be applied. Although the addition of redundant subsystems can often improve both performance and system flexibility, the overall cost of the system cannot be ignored.

Accordingly, a need exists to improve the flexibility of large multiprocessor computer systems.

SUMMARY OF THE INVENTION

The above needs and other advantages, for multiprocessor systems with many resources that may be arranged to support different tasks, are provided by the present inventive method and system where the resources may be applied depending on the number and/or type of I/O devices or device controllers attached to the various I/O ports and/or the number and type of transactions anticipated at the various I/O ports. Such arrangements may increase the processing speed and/or reduce the latency of response or be more efficient in use of main, cache or other memories. In one arrangement the present invention provides the ability to "hot swap" or "hot plug" some electronic sub-assemblies, that is replace the sub-assemblies without removing power from or shutting down the remaining system.

Sub A1

The present invention provides a method for programmably allocating system resources to accommodate I/O transactions at I/O ports of a multiprocessor computer system. The inventive method determines the number and type of transactions anticipated at a port, the number and type of devices being serviced via the port, a criteria for the transactions at the port with respect to the number and type of transactions and device, and assigns the system resources to the port with respect to the criteria. In preferred embodiments the criteria may include, among other parameters, increasing system operating speeds, reducing latency, or ensuring that some devices, even low priority devices, are serviced, slowing the system to allow debugging and other such servicing, ensuring proper communications credits, and supporting hot swapping of processor and module assemblies.

In one embodiment one or more control registers are provided for each port. The control registers may includes a plurality of programmable fields. Where additional control registers are used, the many fields are distributed among the control registers.

The control registers may be configured, in a preferred embodiment, to contain the number of direct memory access engines available at a port to support a transaction, the number of cache lines available for requested data, and a number representing priorities among the anticipated transactions

With respect to hot swapping assemblies, the state of the assembly being replaced, including its associated memory systems and their status and control registers, and the contents of its cache and memory systems is preserved. The remaining system is informed of such swapping with respect to any cached data, and the system directory is updated.

In particular, the local cache and local memory contents are stored so as not to be affected by the hot swapping, the cache data is flushed or invalidated, a flush indicator is set in the cache status and control register, no new transactions are allowed, and any transactions started or pending are completed, translating look-aside buffers are invalidated.

BRIEF DESCRIPTION OF THE DRAWINGS

The above and further advantages of the invention may be better understood by referring to the following description in conjunction with the accompanying drawings, in which like reference numbers indicate identical or functionally similar elements:

Fig. 1, previously discussed, is a schematic block diagram of a conventional symmetrical multiprocessor computer system;

Fig. 2 is a schematic block diagram of a symmetrical multiprocessor computer system in accordance with the present invention;

Fig. 3 is a schematic block diagram of a dual processor module of the computer system of Fig. 2;

Fig. 4 is a schematic block diagram of an I/O bridge in accordance with the present invention;

Fig. 5 is a schematic block diagram of an I/O subsystem of the computer system of Fig. 2;

Fig. 6 is a schematic diagram of an illustrative embodiment of four (4) 8P drawers of the SMP system mounted within a standard 19-inch rack;

Fig. 7 is a functional block diagram of an I/O bridge;

Figs. 8-9 are schematic blocks diagram of various ports of the I/O bridge of Fig. 7;

Sub D1

Figs. 10A-C, 11, and 12 are detailed block diagrams of preferred register formats utilized at the I/O bridge of the present invention;

DETAILED DESCRIPTION OF AN ILLUSTRATIVE EMBODIMENT

Fig. 2 is a schematic block diagram of a data processing system that may advantageously include the present invention. In the illustrative embodiment, the data processing system is preferably a symmetrical multiprocessor (SMP) system 200 comprising a plurality of processor modules 300 interconnected to form a two dimensional (2D) torus configuration. Each processor module 300 comprises two central processing units

(CPUs) or processors 202 and has connections for two input/output (I/O) ports (one for each processor 202) and six inter-processor (IP) network ports. The IP network ports are preferably referred to as North (N), South (S), East (E) and West (W) compass points and connect to two unidirectional links. The North-South (NS) and East-West (EW) compass point connections create a (manhattan) grid, while the outside ends wrap-around and connect to each other, thereby forming the 2D torus. The SMP system 200 further comprises a plurality of I/O subsystems 500. I/O traffic enters the processor modules 300 of the 2D torus via the I/O ports. Although only one I/O subsystem 500 is shown connected to each processor module 300, because each processor module 300 has two I/O ports, any given processor module 300 may be connected to two I/O subsystems 500 (i.e., each processor 202 may be connected to its own I/O subsystem 500).

Fig. 3 is a schematic block diagram of the dual CPU (2P) module 300. As noted, the 2P module 300 comprises two CPUs 202 each having connections 310 for the IP ("compass") network ports and an I/O port 320. The 2P module 300 also includes one or more power regulators 330, server management logic 350 and two memory subsystems 370 each coupled to a respective memory port (one for each CPU 202). The system management logic 350 cooperates with a server management system to control functions of the SMP system 200. Each of the N, S, E and W compass points along with the I/O and memory ports, moreover, use clock-forwarding, i.e., forwarding clock signals with the data signals, to increase data transfer rates and reduce skew between the clock and data.

Each CPU 202 of a 2P module 300 is preferably an "EV7" processor that includes part of an "EV6" processor as its core together with "wrapper" circuitry comprising two memory controllers, an I/O interface and four network ports. In the illustrative embodiment, the EV7 address space is 44 physical address bits and supports up to 256 processors 202 and 256 I/O subsystems 500. The EV6 core preferably incorporates a traditional reduced instruction set computer (RISC) load/store architecture. In the illustrative embodiment described herein, the EV6 core is an Alpha® 21264 processor chip manufactured by Compaq Computer Corporation of Houston, Texas, with the addition of a 1.75 megabyte (MB) 7-way associative internal cache and "CBOX," the latter providing integrated cache controller functions to the EV7 processor. However, it will be apparent to those skilled in the art that other types of processor chips may be advantageously used. The EV7 processor also includes an "RBOX" that provides integrated routing/networking control functions with respect to the compass points, and a "ZBOX" that provides integrated memory controller functions for controlling the memory subsystem 370.

Each memory subsystem 370 may be and/or may include one or more conventional or commercially available dynamic random access memory (DRAM), synchronous DRAM (SDRAM), double data rate SDRAM (DDR-SDRAM) or Rambus DRAM (RDRAM) memory devices. Each EV7 processor 202, moreover, can operate with 0, 1 or 2 memory controllers.

Fig. 4 is a schematic block diagram of an I/O bridge or "IO7" 400, that provides a fundamental building block for the SMP I/O subsystem. The IO7 is preferably implemented as an application specific integrated circuit (ASIC). Each EV7 processor supports one I/O ASIC connection; however, there is no requirement that each processor have an I/O connection. In the illustrative embodiment, the I/O subsystem 500 includes a PCI and/or PCI-X I/O expansion box with hot-swap PCI-x and AGP support. The PCI-x expansion box includes an IO7 plug-in card that spawns four I/O buses.

The IO7 400 comprises a North circuit region 410 that interfaces to the EV7 processor and a South circuit region 450 that includes a plurality of I/O ports 460 (P0-P3) that interface to standard I/O buses. An EV7 port 420 of the North region 410 couples to the EV7 processor via 2 unidirectional, clock forwarded links 430. Each link 430 has a 32-bit data path that operates at 400 Mbps for a total bandwidth of 1.6 GB in each direction. In the illustrative embodiment, three of the four I/O ports 460 interface to the PCI and/or PCI-X bus, while the fourth port interfaces to an AGP bus.

In accordance with an aspect of the present invention, a cache coherent domain of the SMP system 200 extends into the IO7 400 and, in particular, to I/O caches located within each I/O port 460 of the IO7 400. Specifically, the cache coherent domain extends to a write cache (WC) 462, a read cache (RC) 464 and a translation look-aside buffer (TLB) 466 located within each I/O port 460. As described further herein, the caches 462, 464, 466 function as coherent buffers.

Referring again to the embodiment of Fig. 2, the 2D-torus configuration of the SMP system 200 may comprise sixteen EV7 processors 202 interconnected within two 8P drawer enclosures 600. Specifically, there are four 2P modules 300 interconnected by a backplane within each enclosure 600. In the illustrative embodiment, four 8P drawers 600 may be mounted within a standard 19-inch rack (2 meters in height) as shown in Fig. 6. Mounting four 8P drawers 600 in a single rack creates a substantial cabling problem when interconnecting the 32 processors within the 2D-torus configuration and when coupling the processors to the I/O subsystems 500 via the IO7 devices 400 associated with the EV7 processors 202. Note that multiple racks may be cabled together to net a system of 64, 128, 256 or a larger number of processors. In accordance with a related invention, an efficient means for interconnecting cables among the 8P drawers 600 of a fully-configured, 19-inch rack is provided.

Fig. 5 is a schematic block diagram of an I/O subsystem or drawer 500 of the SMP system 200. Each I/O subsystem 500 includes a first I/O riser card 510 containing an IO7 400, a connector 520 coupling the IO7 400 to its EV7 processor 202 and a plurality of I/O buses. The speed of the I/O buses contained within the I/O subsystem 500 is a function of the length and the number of loads of each I/O bus. The I/O subsystem 500 is divided into two parts: a hot-plug region 530 and an embedded region 550. In the illustrative embodiment, there is a dedicated slot 560 adjacent to the I/O riser card 510 within the embedded region 550 that is dedicated to a 4x AGP Pro graphics card. Additional slots (e.g., for power and an additional data path) may be provided to support the AGP Pro card. Also included within the embedded region 550 are three standard, 64-bit PCI card slots 572-576, which are available for embedded I/O card options. For example, an I/O standard module card 580 may be inserted within one of the PCI card slots 572-576.

Each I/O subsystem 500 also includes power supplies, fans and storage/load devices (not shown). The I/O standard module card 580 contains a Small Computer System Interface (SCSI) controller for storage/load devices and a Universal Serial Bus (USB) that enables keyboard, mouse, CD and similar input/output functions. The embedded region 550 of the I/O subsystem 500 is typically pre-configured and does not support hot-swap operations. In contrast, the hot-plug region 530 includes a plurality of slots adapted to support hot-swap. Specifically, there are two ports 532, 534 of the hot plug region 530 dedicated to I/O port one (P1 of Fig. 4) and six slots 538-548 dedicated to I/O port two (P2). Likewise, the dedicated AGP Pro slot 560 comprises port three (P3) and the three embedded PCI slots 572-576 comprise port zero (P0). The I/O buses in the hot-plug region 530 are configured to support PCI and/or PCI-X standards operating at 33 MHz, 50 MHz, 66 MHz, 100 MHz and/or 133 MHz. Not all slots are capable of supporting all of these operating speeds.

Also included within the I/O subsystem 500 and coupled adjacent to the IO7 400 is a PCI backplane manager (PBM) 502. The PBM 502 is part of a platform management infrastructure. The PBM 502 is coupled to a local area network (LAN), e.g., 100 base T LAN, by way of another I/O riser board 590 within the I/O drawer 500. The LAN provides an interconnect for the server management platform that includes, in addition to the PBM 502, a CPU Management Module (CMM) located on each 2P module 300 (Fig. 3) and an MBM (Marvel Backplane Manager).

Fig. 7 is a schematic block diagram of an IO7 400 in greater detail. As shown, the South region 450 of IO7 400 includes four data south ports, which may be numbers SP0-SP3. In addition to the read buffer, write buffer and TLB, as described above, ports SP0-SP3 further include an up hose ordering engine (UPE), a down hose ordering engine (DNE), a down hose forward initiator (DFI), a down hose address buffer (DNA), and a control and status register (CSR) block. South ports SP0-SP2, which may be configured to run the PCI and/or PCI-X bus standards, further include hot plug interface gates (HPIG).

Fig. 8 is a representative block diagram of one of south ports SP0-SP2, each of which preferably includes a PCI or PCI-X controller for controlling the PCI or PCI-X bus to which the port SP0-SP2 is coupled. As shown, the UPE of south ports SP0-SP2 has a plurality (preferably twelve) DMA controllers, which are numerically identified “00” to “11.”

Fig. 9 is a representative block diagram of South port SP3 which, as described above, is configured to support AGP. Fig. 10 is a representative block diagram of the interrupt port P7 of South region 450, and Fig. 11 is a block diagram of the MUX of South region 450.

As described above, each port P0-P3 of the IO7 has a plurality of DMA engines/state machines (preferably twelve) that are used to process transactions for the I/O devices coupled to that port. In accordance with another aspect of the present invention, a method is disclosed for tuning the availability of resources needed to process I/O transactions.

Sub A2 In particular, for each port P0-P3 of the IO7 there is a POx_CTRL control register 800 of Fig. 7 having a plurality of fields. Figs. 11A-C are a detailed preferred format of a the POx_CTRL control register 1500. A POx_CTRL control register 800 is preferably disposed at each port P0-P3 of the IO7 and is read during initialization of the IO7. It may also be read during operation of the IO7. As shown, the POx_CTRL control register format 1500 is organized into a plurality of fields. An RM_TYPE field 1502 (Fig. 10C) is preferably used, at least in part, to control a novel pre-fetch algorithm, which is disclosed in a co-pending patent serial no.________________, entitled, Adaptive Data Prefetch Prediction Algorithm, filed ____________________. Which application is hereby incorporated herein by reference. In particular, the RM_TYPE field 1502 controls the maximum number of DMA engines that may be assigned to process a given transaction. The RM_TYPE field 1502 may be 2-bits long.

For example, if the RM_TYPE field 1502 is set to "00", then the port may assign at most two DMA engines to any given transaction. If the RM_TYPE field 1502 is set to "01", then up to six DMA engines may be assigned to a single transaction. If the RM_TYPE field 1502 is set to "10", up to eight DMA engines may be assigned to a single transaction. If it is set to "11", up to eleven DMA engines may be assigned to a single transaction.

It should be understood that other values may also be assigned and/or additional bits used to provide finer granularity.

Those skilled in the art will recognize that, by adjusting the RM_TYPE field 1502, each port P0-P3 of an IO7 may be individually tuned for the type and number of transactions that are anticipated on that port. For example, if many I/O devices are coupled to a given port, then the RM_TYPE field 1502 may be set to "00" to ensure that no one I/O device ends up consuming all of the DMA engines for its own transaction(s). In contrast, if only a single I/O device is coupled to a given port, the RM_TYPE field 1502 may be set to "11" so that the one I/O device can take advantage of the DMA engine resources at that port.

The IO7 has a certain number of miss addressed file (MAF) values that may be used to request data (in terms of cache lines) from the SMP system 200 in response to a cache line miss at the IO7. In accordance with another aspect of the present invention, the number of MAF values at a given IO7 is user programmable. More specifically, each IO7 preferably includes a IO7_MAF register. The IO7_MAF register may be 5-bits in length. By setting the IO7_MAF register, a user may adjust the number of MAF values available to the IO7. For example, if the IO7_MAF register is set to "0000", then no MAF values are available to the IO7. If the IO7_MAF register is set to "0001", then one MAF value is available. If the IO7_MAF register is set to "0010", then two MAF values are available, and so on up to a maximum of 31 MAF values. The IO7_MAF register basically provides a mechanism by which a user can "throttle" the performance of a selected IO7. This may be desirable to perform debugging and other service related tasks, as well to control the operation of the IO7s. Preferably, at least two MAF values are provided to the IO7.

Sub A3

The IO7 400 utilizes a credit-based flow control system to communicate with its respective EV7 processor 202. In particular, for each of the Read I/O, Write I/O, Request, Block Response and No Block Response virtual channels, the MUX has a corresponding credit buffer. If the MUX has a packet to be transmitted on the Request channel, it first checks to see if there is at least one credit in the Request channel credit buffer against which to "charge" this packet. If a credit exists, then the IO7 assumes that the EV7 processor 202 has sufficient buffer space to store the packet. Accordingly, the MUX decrements the Request channel credit buffer by "1" (i.e., the number of messages to be sent) and sends the message. If there are no Request channel credits, the MUX

A3 contd.

must wait until at least one Request credit is received from the EV7 processor 202 before sending the message.

In accordance with another aspect of the present invention, the starting number of credits for the Read I/O, Write I/O, Request, Block Response and No Block Response virtual channels is preferably programmable by writing to a corresponding register at the IO7. In particular, the IO7 preferably includes an IO7_UPH control register.

Fig. 11 is a detailed format of a preferred IO7_UPH control register 1600. The IO7_UPH control register 1600 includes a plurality of fields, including a plurality of up hose credit fields 1602. In particular, the IO7_UPH control register 1600 has an up hose Request channel credit field 1602a, an up hose Read I/O channel credit field 1602b, an up hose Write I/O channel credit field 1602c, an up hose Block Response channel credit field 1602d, and an up hose No Block Response credit field 1602e. Each of the credit fields 1602 is programmable so that a user may independently set the number of credits available for each channel. Each credit field 1602 may be 5-bits in length. Accordingly, each channel may be programmed with "0" to "31" credits.

By adjusting the maximum number of credits available for each of these virtual channels, a user can finely tune the operation of each IO7 within the SMP system 200.

CPU Hot Plug

As described above, an EV7 processor 202 and/or the corresponding 2P module 300 can be hot swapped. That is, a selected EV7 processor 202 and/or 2P module 300 can be removed and replaced with a new processor or module without bringing the entire SMP system 200 (or the partition in which the processor or module is located) down. Prior to removing an EV7 processor 202, several steps must be performed. Specifically, all data in the EV7 processor's local cache and in the memory subsystem 370 directly controlled by the EV7 processor 202 must be "flushed" to secondary (or alternative) storage. The information contained in the respective directory (e.g., cache line ownership status and location) must also be flushed. Additionally, new transactions that target the EV7 processor 202 or that transit through it must be stopped and/or re-routed.

As described above, the transactions initiated by a given IO7 are received, at least initially, by the EV7 processor 202 to which the IO7 400 is coupled. In accordance with an aspect of the present invention, an efficient system for stopping the transactions initiated by a given IO7 400 in order to facilitate hot swap of the respective EV7 processor 202 is described.

Sub A4 Specifically, as described above, each IO7 400 includes a POx_CTRL control register 1500 (Figs. 11A-C) that contains information utilized by the IO7 400 when it is initialized. The POx_CTRL register 1500 preferably includes a UPE_ENG_EN field 1504 (Fig. 10C). The UPE_ENG_EN field 1504 preferably includes at least one bit for each DMA engine at the IO7 400. In the preferred embodiment, each IO7 400 has twelve DMA engines. Accordingly, the UPE_ENG_EN field 1504 has twelve DMA engine enable bits. When the IO7 400 is initialized, it reads the POx_CTRL control register 1500, including the UPE_ENG_EN field 1504 and, among other things, starts and runs a DMA engine for each DMA engine enable bit that is asserted. In this way a user can program the number of DMA engines (up to some maximum, e.g., twelve) that are started and run at a given IO7 400. If an EV7 processor 202 is to be hot swapped a user, operating through system software or firmware, preferably de-asserts all twelve DMA engine enable bits of the IO7 400 coupled to the EV7 202 that is to be removed. That is, the user sets all bits of the UPE_ENG_EN field 1504 of the respective POx_CTRL control register 1500 to "0". In response, the IO7 400 stops allocating DMA engines for new transactions, thereby stopping the IO7 400 from commencing new transactions. When a DMA engine that was in use is subsequently disabled, it nonetheless completes the pending or existing transaction(s) that were assigned to it.

Sub A5 In addition to stopping the IO7 400 from initiating any new transactions by disabling its DMA engines, the user also causes any data stored in the IO7's WCs 462, RCs 464 and TLBs 466 to be invalidated, whether that data is coherent or not. To facilitate this operation, among other reasons, the IO7 400 further includes a POx_CACHE_CTL register at each port 460, which governs the operation of the WC 462 and RC 464 at that port 460. Fig. 12 is a schematic block diagram of a preferred format of a POx_CACHE_CTL register 1700. The POx_CACHE_CTL register 1700 includes a

A5 Cont'd.

UPE_FLUSH_CACHE field 1702, which may be 1-bit. If asserted, the UPE_FLUSH_CACHE field 1702 causes the IO7 400 to flush all coherent and non-coherent data stored in the WC 462 and RC 464 for that port 460. Accordingly, as part of the hot swapping of an EV7 processor 202, the user also asserts the flush bit of each port's cache status and control register. In response, the IO7 400 invalidates the contents of all of its WCs 462 and RCs 464; and for each entry of the WCs 462 and RCs 464, the IO7 400 returns Victim or VictimClean messages to the directory depending, among other things, on the ownership status of the invalidated data.

Sub A6

For each of its TLBs 466, the IO7 400 preferably includes a translation invalidate all (TBIA) register. If the TBIA register contains any value, including all zeros, the IO7 400 preferably responds by flushing the contents of the respective TLB 466. Accordingly, as part of the hot swapping of an EV7 processor 202, the user also writes any value to each TBIA register of the affected IO7 400. In response, the IO7 400 invalidates the contents of all of its TLBs and sends VictimClean messages to the directory 380.

The POx_CACHE_CTL register 1700 at each port 460 the IO7 400 also includes a UPE_CACHE_INVAL field 1704 which indicates whether one or more blocks of the WC 462, RC 464 or TLB 466 of that port 460 contain valid data. After invalidating the contents of the port's WC 462, RC 464 and TLB 466, as described above, the IO7 400 preferably de-asserts the UPE_CACHE_INVAL field 1704. Before actually removing the EV7 processor 202 that is to be hot-swapped, the user preferably confirms that the UPE_CACHE_INVAL field 1704 is de-asserted. It should be understood that the state of the EV7 processor 202 to be removed (and the state of its associated memory subsystem 370) is copied to a new location before the EV7 processor 202 is removed.

What is claimed is: